Código Limpo e seu Mapeamento em Métricas de Código-fonte

Lucianna T. Almeida
João Machini

Orientador: Fabio Kon
Coorientador: Paulo Meirelles

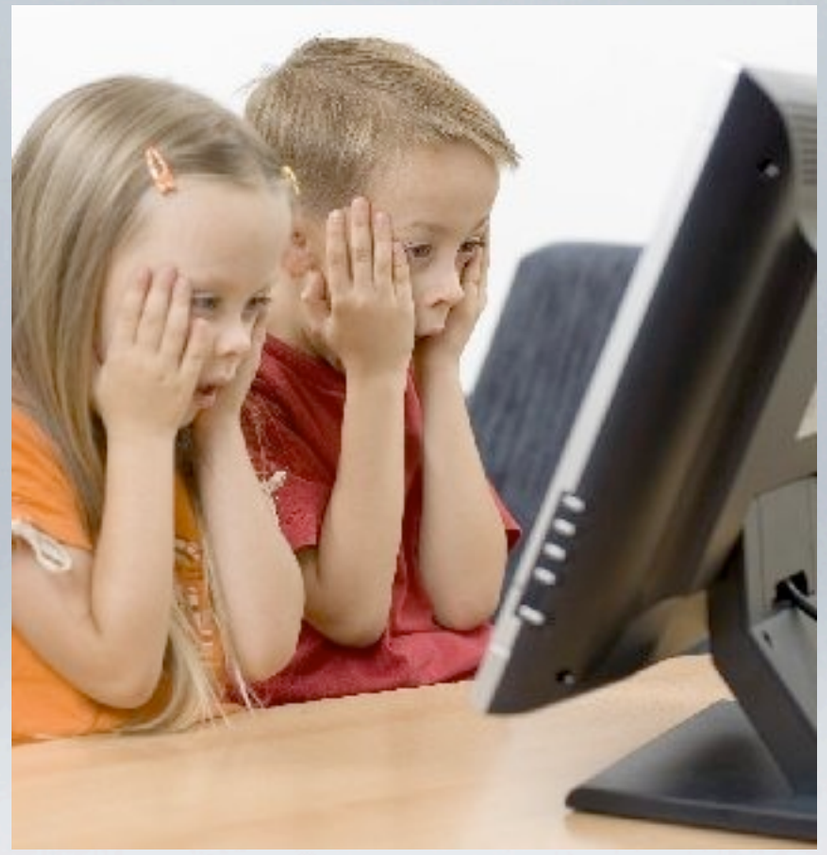
O Problema

```
void dslk(unsigned char janela[],int *posprim,
                int *posult,int primeira,int aux,
                int dist,unsigned char compr,int n)
{
     int i,t;
     int auxi;
     t=0;
     auxi=n;
     primeira=primeira-n;
     if(dist==0 && aux < 2){
          (*posprim)--;
          for(i=(*posprim);i<254;i++)
               janela[i]=janela[i+1] + dlk2((*posprim));
          janela[254]=janela[256];
     }
     else{
          (*posprim)=(*posprim)-compr;
          if(*posprim<0)
               (*posprim)=0;
          for(;auxi>0;auxi--)
               for(t=(*posprim);t<254;t++)
                    janela[t]=janela[t+1];
          aux=aux-n;
          for(;n>0;n--)
               janela[255-n]=janela[(aux-n)+1];
     }
}
```

Problema

```
void dslk(unsigned char janela[],int *posprim,
                int *posult,int primeira,int aux,
                int dist,unsigned char compr,int n)
{
     int i,t;
     int auxi;
     t=0;
     auxi=n;
     primeira=primeira-n;
     if(dist==0 && aux <
          (*posprim)--;
          for(i=(*posprim)
               janela[i]=janela[i+1] + dlk2((*posprim));
          janela[254]=janel[256];
     }
     else{
          (*posprim)=(*posprim)-compr;
          if(*posprim<0)
               (*posprim)=0;
          for(;auxi>0;auxi--)
               for(t=(*posprim);t<254;t++)
                    janela[t]=janela[t+1];
          aux=aux-n;
          for(;n>0;n--)
               janela[255-n]=janela[(aux-n)+1];
     }
}
```

O Problema



Código Limpo

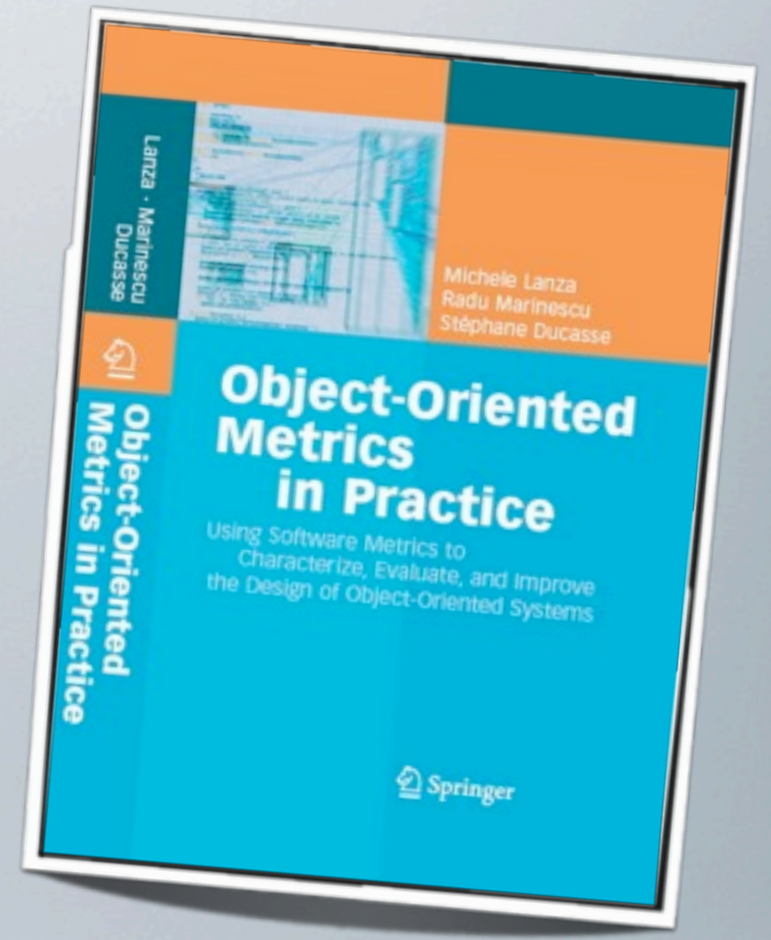
Orientação a Objetos

Simplicidade

Flexibilidade

Expressividade

Mapeamento



Métricas de
Código-fonte

\+

Cenários

=

Ajuda na Detecção
de Problemas

Nosso TCC

Métricas

Código

Conceitos

Código
Limpo

Métodos

Métodos

```
def custosAPartirDoVertice(vertice):
    custos = novo Lista(numeroDeVertices)
    fila = nova FilaDePrioridades(numeroDeVertices)

    for i in (1, numeroDeVertices):
        custos[i] = -1

    custos[vertice] = 0
    fila.insere(nova Aresta(0,0))

    while(fila.vazia()):
        verticeDoMomento = fila.verticeDaArestaComCustoMinimo()
        for aresta in (arestasDoVertice(verticeDoMomento)):
            verticeDestino = aresta.verticeDestino()
            custo = aresta.custo()

            if(custos[verticeDestino] == -1):
                custos[verticeDestino] = custos[verticeDoMomento] + custo
                fila.insere(nova Aresta(verticeDestino, custos[verticeDestino]))

            else if(custos[verticeDestino] > custos[verticeDoMomento] + custo):
                custos[verticeDestino] = custos[verticeDoMomento] + custo

    return custos
```

Métodos

Alto LOC

```
  def custosAPartirDoVertice(vertice):
1       custos = novo Lista(numeroDeVertices)
2       fila = nova FilaDePrioridades(numeroDeVertices)

3       for i in (1, numeroDeVertices):
4           custos[i] = -1

5       custos[vertice] = 0
6       fila.insere(nova Aresta(0,0))

7       while(fila.vazia()):
8           verticeDoMomento = fila.verticeDaArestaComCustoMinimo()
9           for aresta in (arestasDoVertice(verticeDoMomento)):
10              verticeDestino = aresta.verticeDestino()
11              custo = aresta.custo()

12              if(custos[verticeDestino] == -1):
13                  custos[verticeDestino] = custos[verticeDoMomento] + custo
14                  fila.insere(nova Aresta(verticeDestino, custos[verticeDestino]))

15              else if(custos[verticeDestino] > custos[verticeDoMomento] + custo):
16                  custos[verticeDestino] = custos[verticeDoMomento] + custo

17      return custos
```

Métodos

Alto LOC

Alta CYCLO

```
  def custosAPartirDoVertice(vertice):
1       custos = novo Lista(numeroDeVertices)
2       fila = nova FilaDePrioridades(numeroDeVertices)

3       for i in (1, numeroDeVertices):
4           custos[i] = -1

5       custos[vertice] = 0
6       fila.insere(nova Aresta(0,0))

7       while(fila.vazia()):
8           verticeDoMomento = fila.verticeDaArestaComCustoMinimo()
9           for aresta in (arestasDoVertice(verticeDoMomento)):
10              verticeDestino = aresta.verticeDestino()
11              custo = aresta.custo()

12              if(custos[verticeDestino] == -1):
13                  custos[verticeDestino] = custos[verticeDoMomento] + custo
14                  fila.insere(nova Aresta(verticeDestino, custos[verticeDestino]))

15              else if(custos[verticeDestino] > custos[verticeDoMomento] + custo):
16                  custos[verticeDestino] = custos[verticeDoMomento] + custo

17      return custos
```

Métodos

```
  def custosAPartirDoVertice(vertice):
1       custos = novo Lista(numeroDeVertices)
2       fila = nova FilaDePrioridades(numeroDeVertices)

3       for i in (1, numeroDeVertices):
4           custos[i] = -1

5       custos[vertice] = 0
6       fila.insere(nova Aresta(0,0))

7       while(fila.vazia()):
8           verticeDoMomento = fila.verticeDaArestaComCustoMinimo()
9       ⟶  for aresta in (arestasDoVertice(verticeDoMomento)):
10              verticeDestino = aresta.verticeDestino()
11              custo = aresta.custo()

12      ⟶      if(custos[verticeDestino] == -1):
13                  custos[verticeDestino] = custos[verticeDoMomento] + custo
14                  fila.insere(nova Aresta(verticeDestino, custos[verticeDestino]))

15      ⟶      else if(custos[verticeDestino] > custos[verticeDoMomento] + custo):
16                  custos[verticeDestino] = custos[verticeDoMomento] + custo

17      return custos
```

Alto LOC

Alta CYCLO

Alto MaxNesting

Métodos

```
  def custosAPartirDoVertice(vertice):
1       custos = novo Lista(numeroDeVertices)
2       fila = nova FilaDePrioridades(numeroDeVertices)

3       for i in (1, numeroDeVertices):
4           custos[i] = -1

5       custos[vertice] = 0
6       fila.insere(nova Aresta(0,0))

7       while(fila.vazia()):
8           verticeDoMomento = fila.verticeDaArestaComCustoMinimo()
9       ⟶  for aresta in (arestasDoVertice(verticeDoMomento)):
10              verticeDestino = aresta.verticeDestino()
11              custo = aresta.custo()

12      ⟶      if(custos[verticeDestino] == -1):
13                  custos[verticeDestino] = custos[verticeDoMomento] + custo
14                  fila.insere(nova Aresta(verticeDestino, custos[verticeDestino]))

15      ⟶      else if(custos[verticeDestino] > custos[verticeDoMomento] + custo):
16                  custos[verticeDestino] = custos[verticeDoMomento] + custo

17      return custos
```

Alto LOC

Alta CYCLO

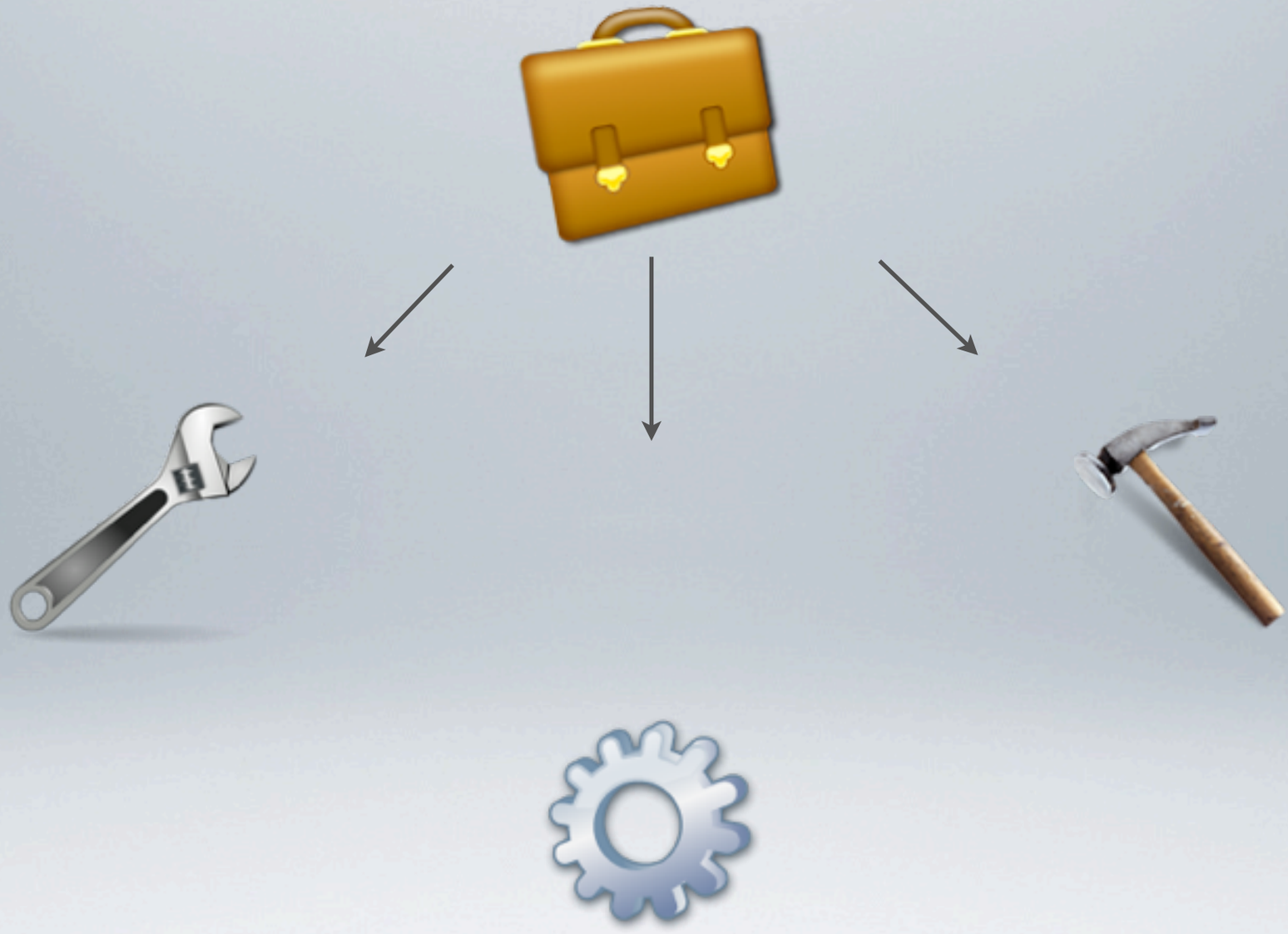
Alto MaxNesting

Muitas tarefas

Muitos detalhes

Métodos

Métodos



Métodos

```
def custosAPartirDoVertice(vertice):
    inicializaCustos()
    inicializaFila()
    atualizaCustosAteAcabarVertices()
```

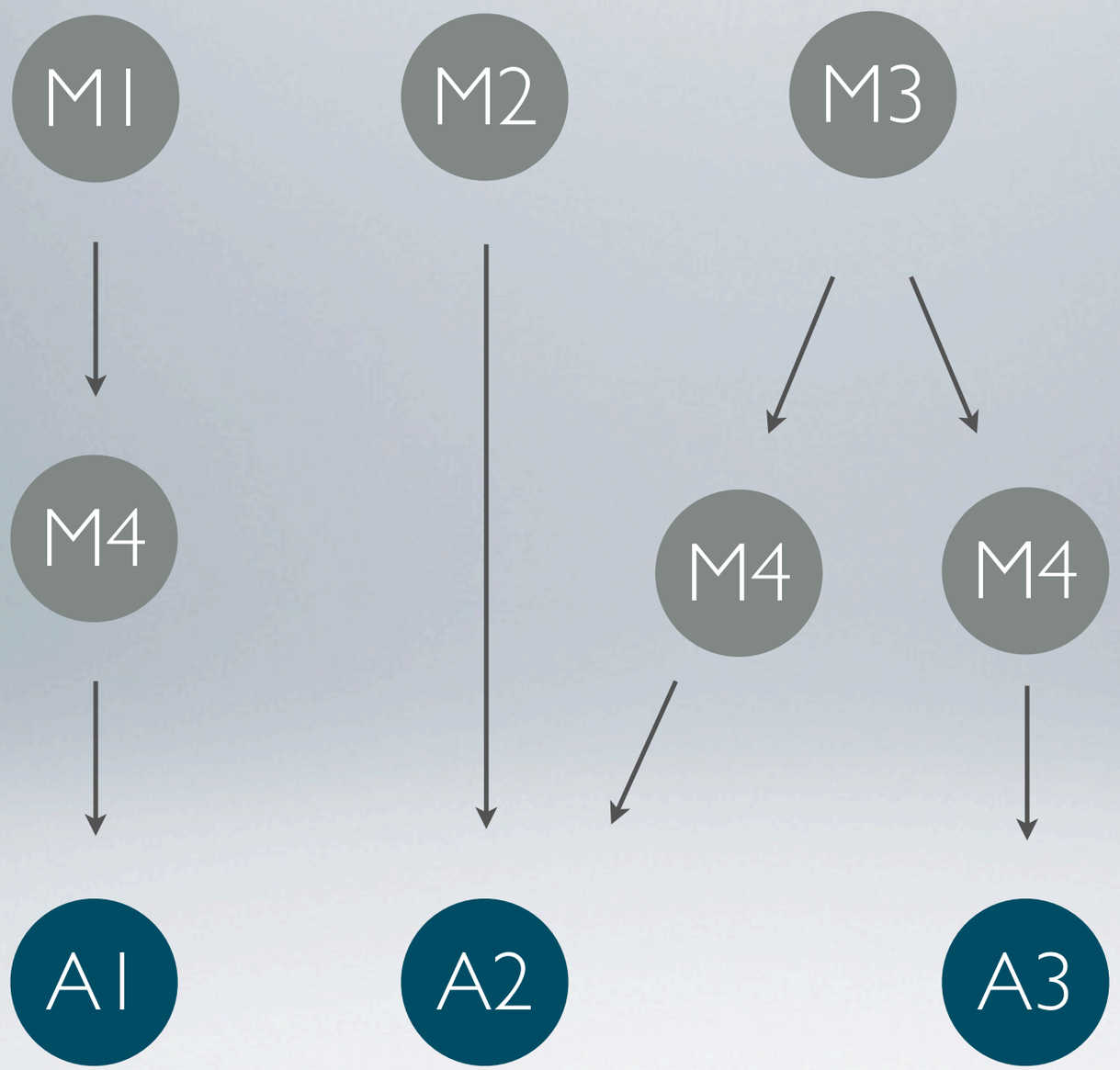
```
def inicializaCustos(vertice):
    for i in (1, numeroDeVertices):
        custos[i] = -1
    custos[vertice] = 0
```

```
def inicializaFila(vertice):
    fila = nova FilaDePrioridades()
    fila.insere(nova Aresta(0,0))
```

```
def atualizaCustosAteAcabarVertices():
    while(fila.vazia()):
        verticeDoMomento = fila.verticeDaArestaComCustoMinimo()
        atualizaCustosAPartirDe(vertice)
```

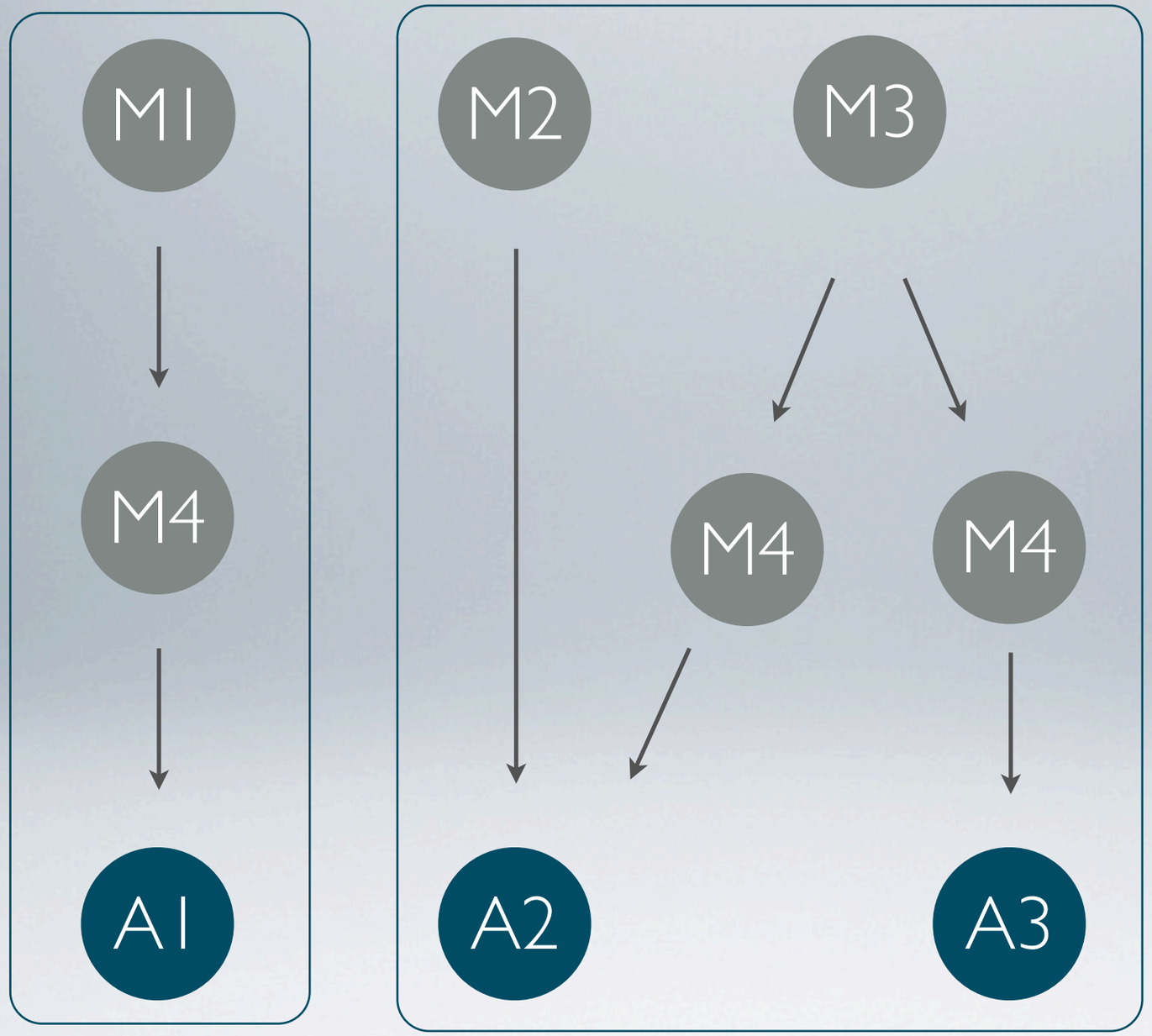
Classes

Classes

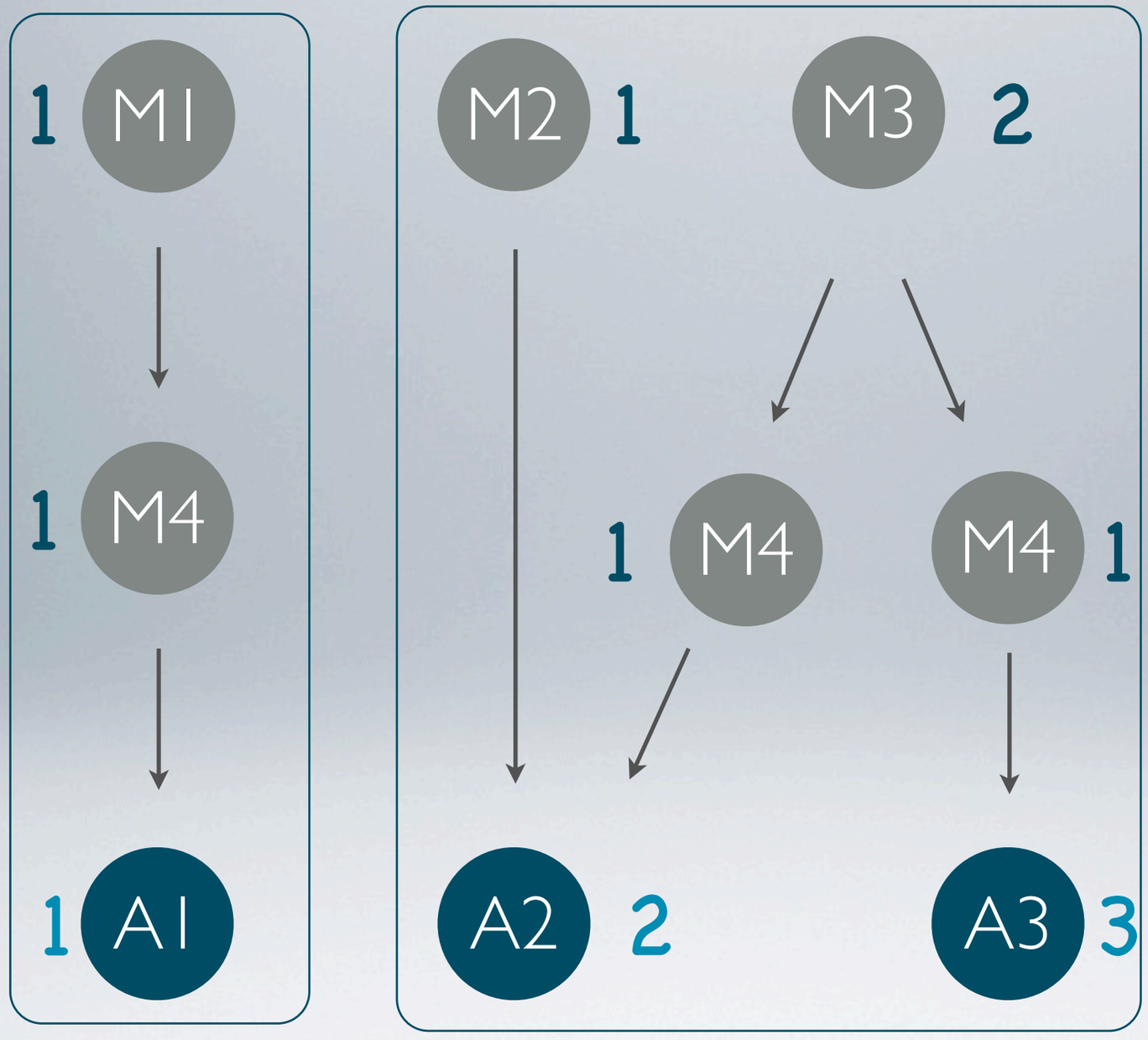


Classes

Alto LCOM4



Classes

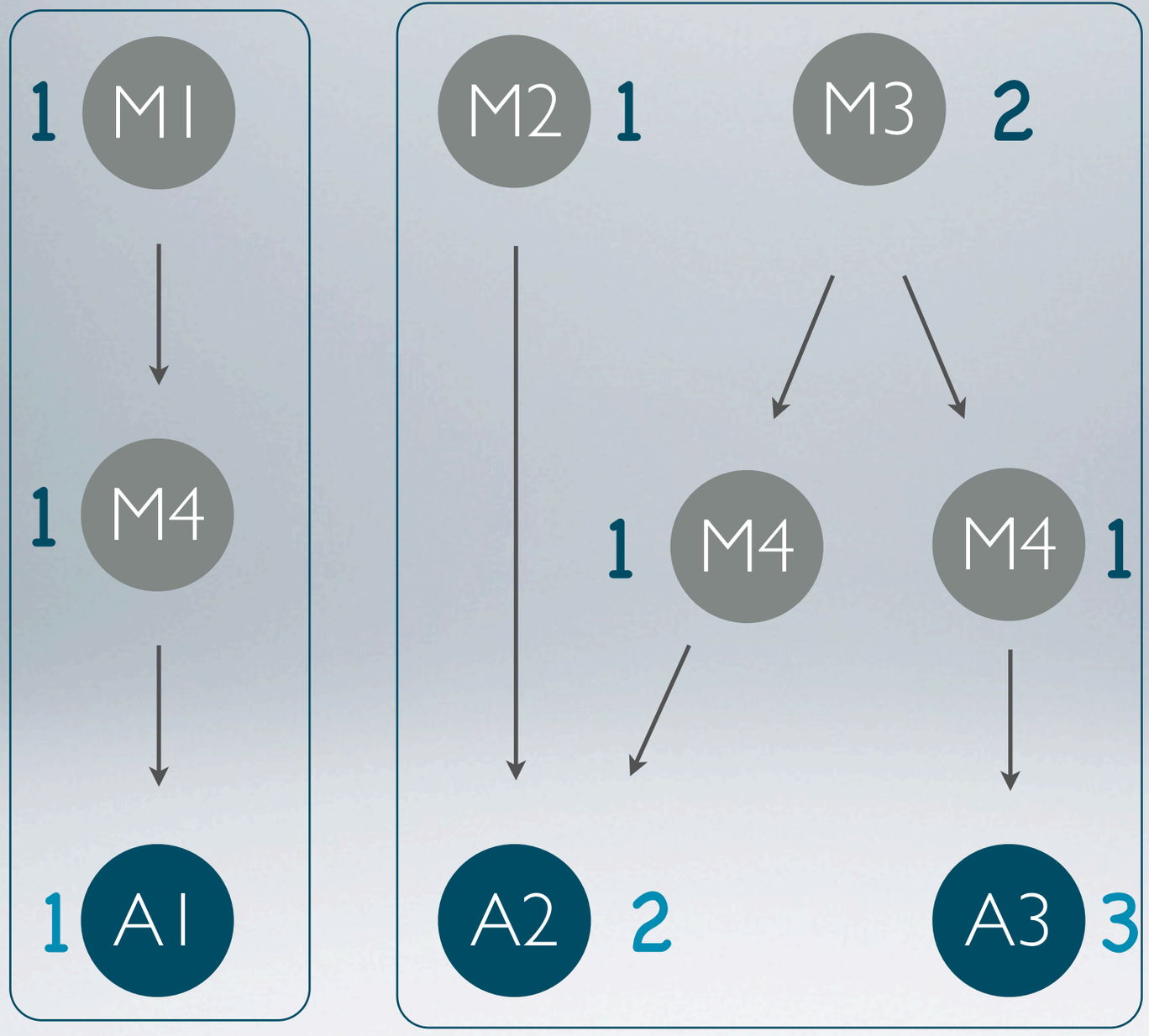


Alto LCOM4

AvgNRA <<< NOA

7/6 = 1.16 3

Classes



Alto LCOM4

AvgNRA <<< NOA

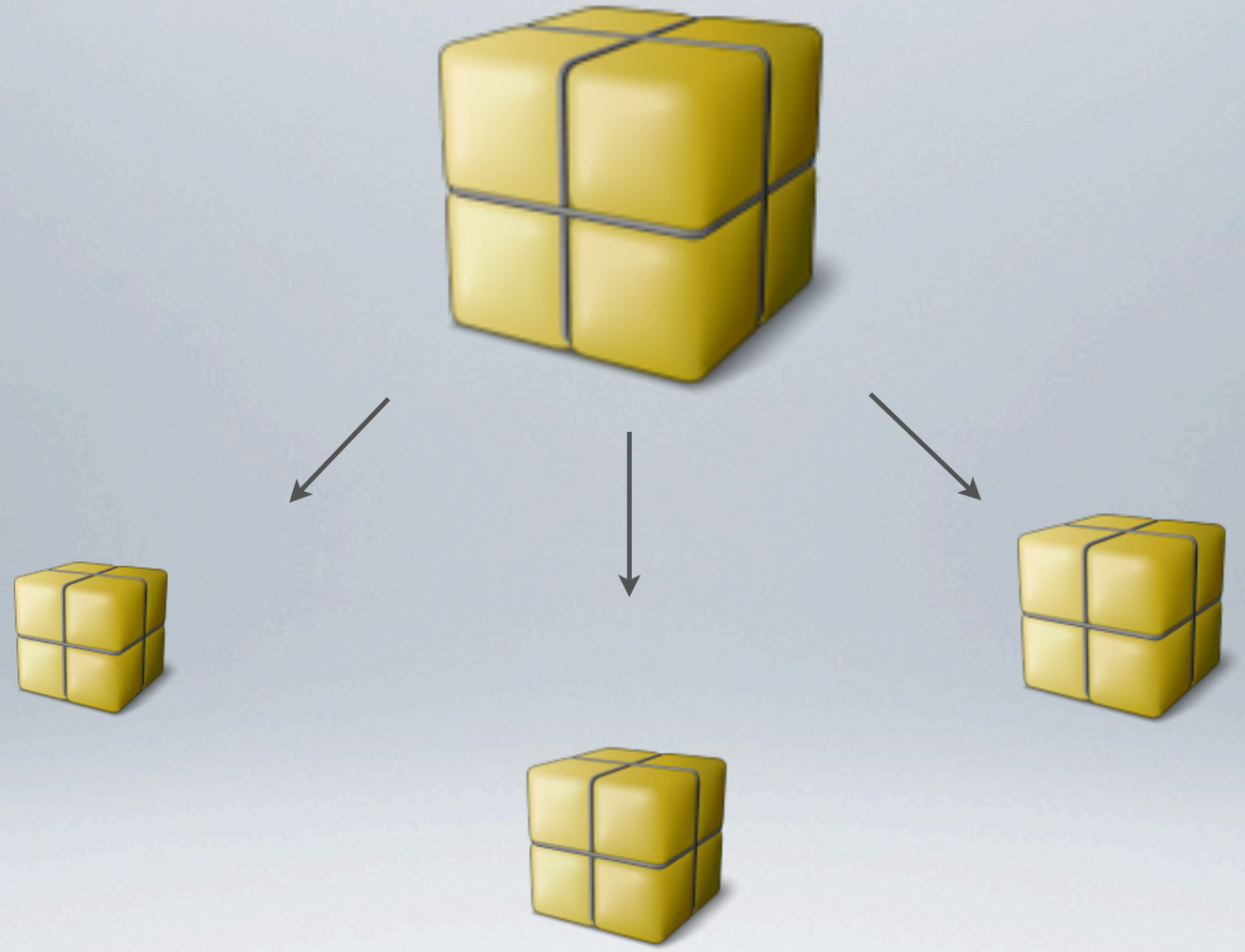
7/6 = 1.16 3

Muitas responsabilidades = Muitas razões para mudar

Classes


1 11 21 31 41 51 61 71 81
2 12 22 32 42 52 62 72
3 13 23 33 43 53 63 73 83
4 14 24 34 44 54 64 74 84
5 15 25 35 45 55 65 75 85
6 16 26 36 46 56 66 76 86
7 17 27 37 47 57 67 77 87
8 18 28 38 48 58 68 78 88
9 19 29 39 49 59 69 79 89
10 20 30 40 50 60 70 80 90

Classes



Estudo de Caso

Analizo-Metrics

- Ferramenta para cálculo de métricas de código-fonte em C++, Java e C

- Software Livre que já colaborávamos

Estudo de Caso

- Melhorias no código da ferramenta para deixá-lo mais limpo

- Observação das métricas durante o desenvolvimento

Referências

- BECK, Kent. *Implementation Patterns*. Addison Wesley, 2007

- MARTIN, Robert C. *Clean Code*, Prentice Hall, 2008;

- LANZA, Michele; MARINESCU, Radu. *Object Oriented Metrics in Practice*. Springer, 2006.


CLEAN CODE
CLEAN CODE
CLEAN CODE